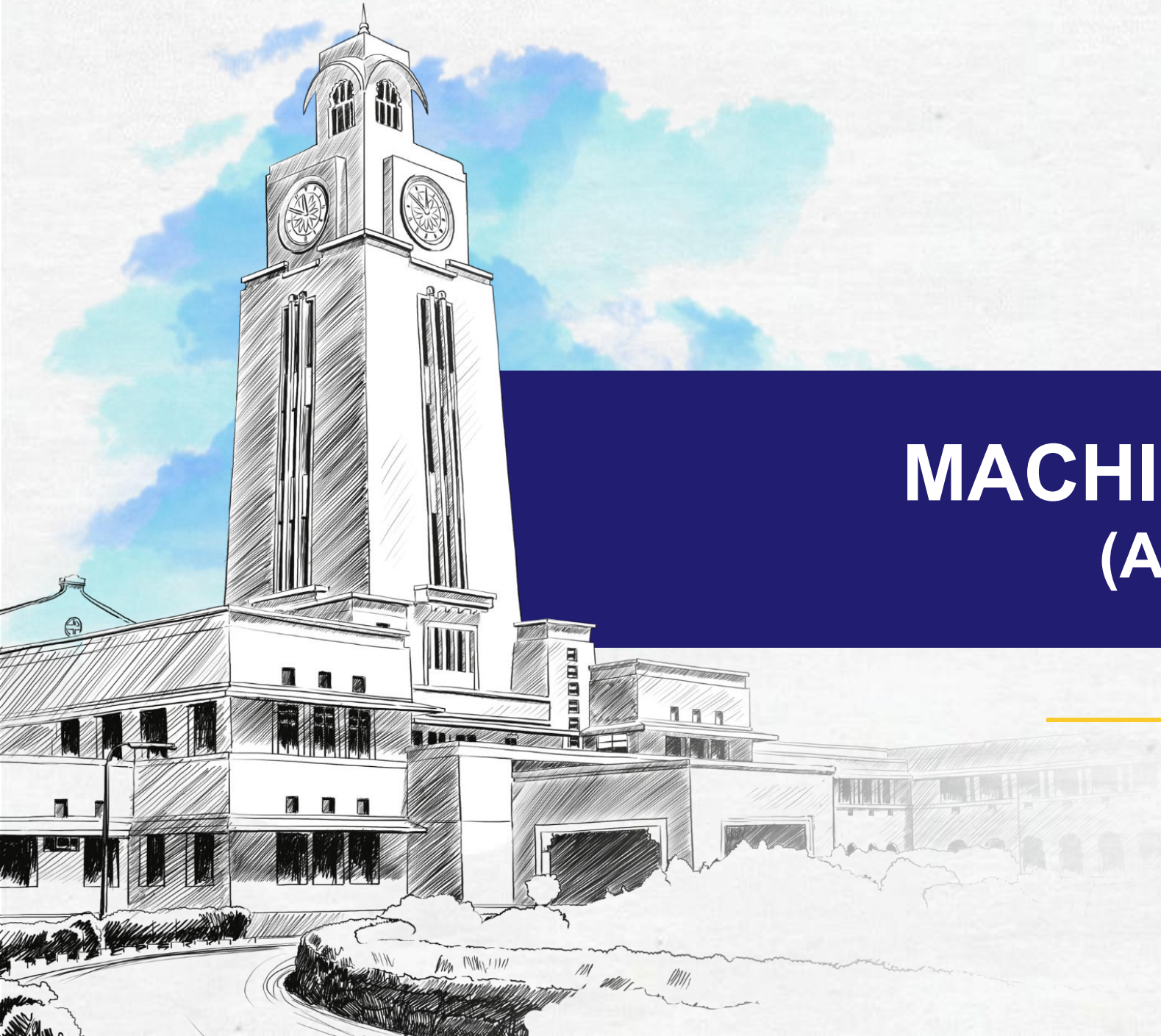


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

# MACHINE LEARNING
## (AIML ZG565)

# Session 10
# (12th April,2026)

# Course Plan

| | |
|---|---|
| M1 | Introduction |
| M2 | Machine learning Workflow |
| M3 | Linear Models for Regression |
| M4 | Linear Models for Classification |
| M5 | Decision Tree |
| M6 | Instance Based Learning |
| M7 | Support Vector Machine |
| M8 | Bayesian Learning |
| M9 | Ensemble Learning |
| M10 | Unsupervised Learning |
| M11 | Machine Learning Model Evaluation/Comparison |

# Classification

**Classification**

- Output is discrete e.g. f(x) : {yes, no, maybe}
- f(x) : {0, 1} or {yes, no}  Binary Classification
- Objective: minimize mis-classification rate

Class label is available → Supervised learning

**Two kinds of Classifiers:**

- Probabilistic Generative Classifiers
  - Models aim to learn joint probability distribution P(X, Y)

Naive Bayes classifier

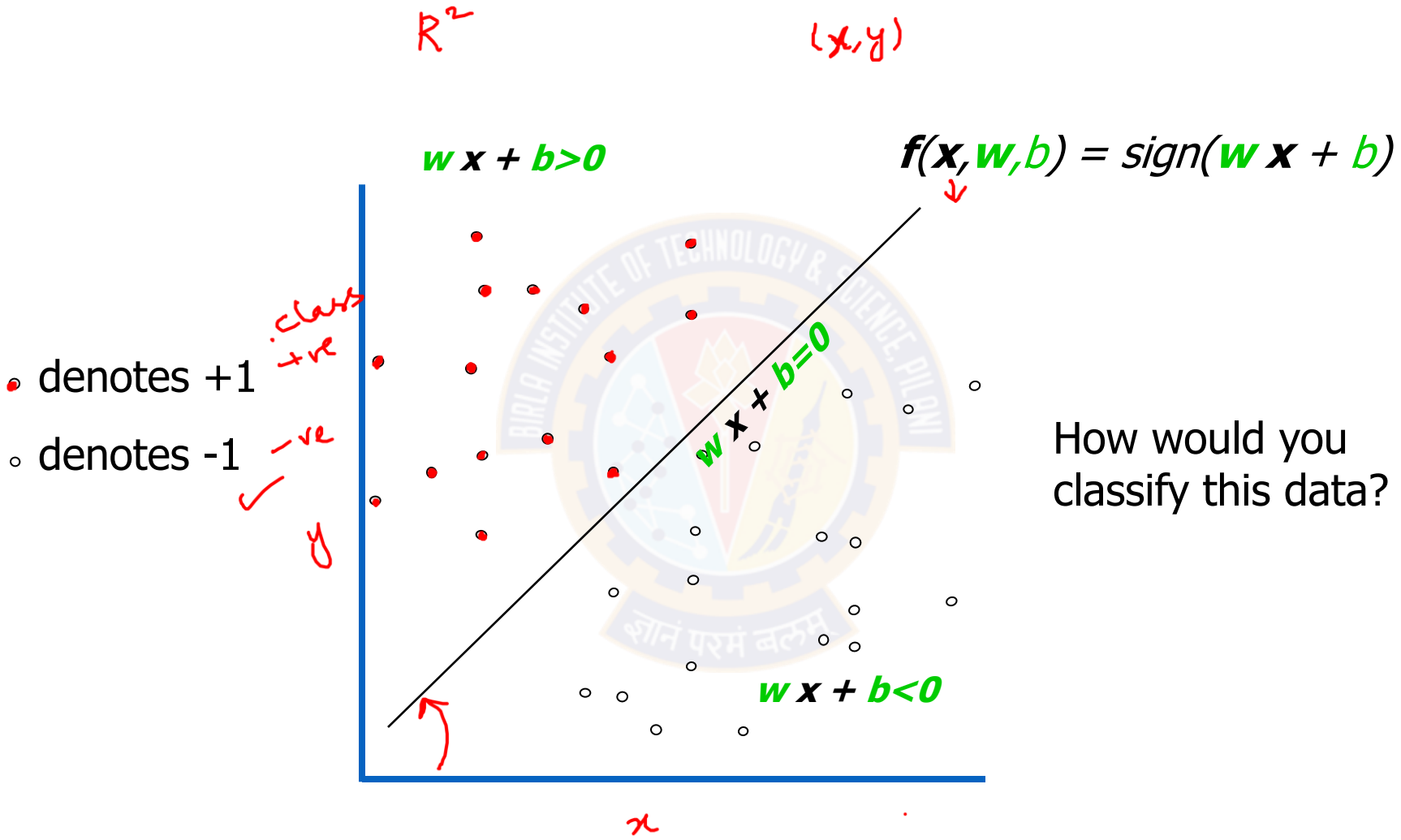
- Probabilistic Discriminative Classifiers
  - Models learn the decision boundary that separate the two classes

Neural Network, SVM

# Agenda

- Linear Classifiers
- Maximum Margin Classification
- Linear SVM
- Soft Margin SVM
- Nonlinear SVM
- Multi-Class Problem

# Linear Classifiers

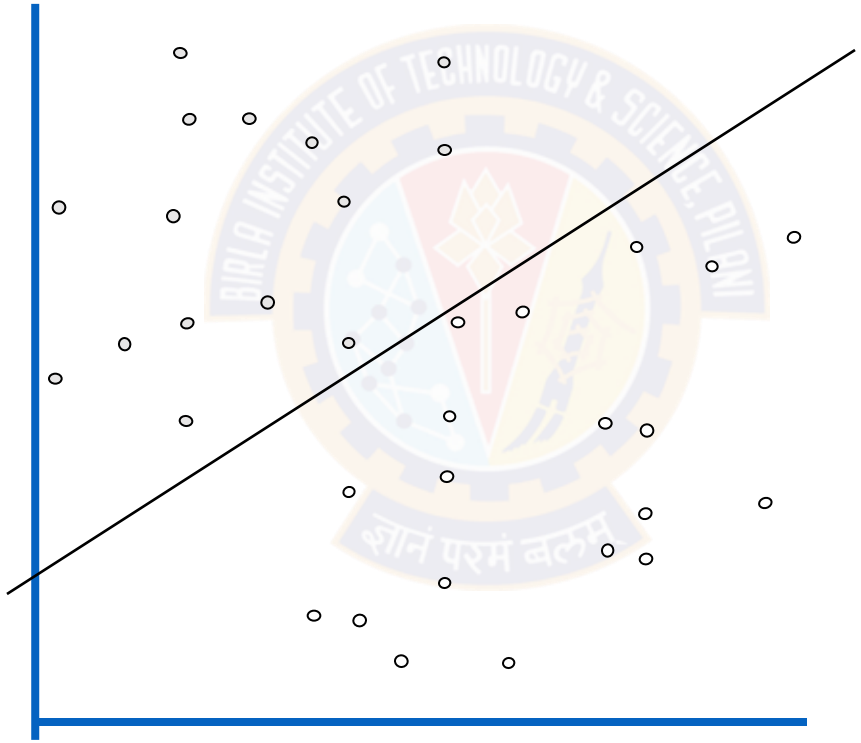



How would you classify this data?

# Linear Classifiers

$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$$

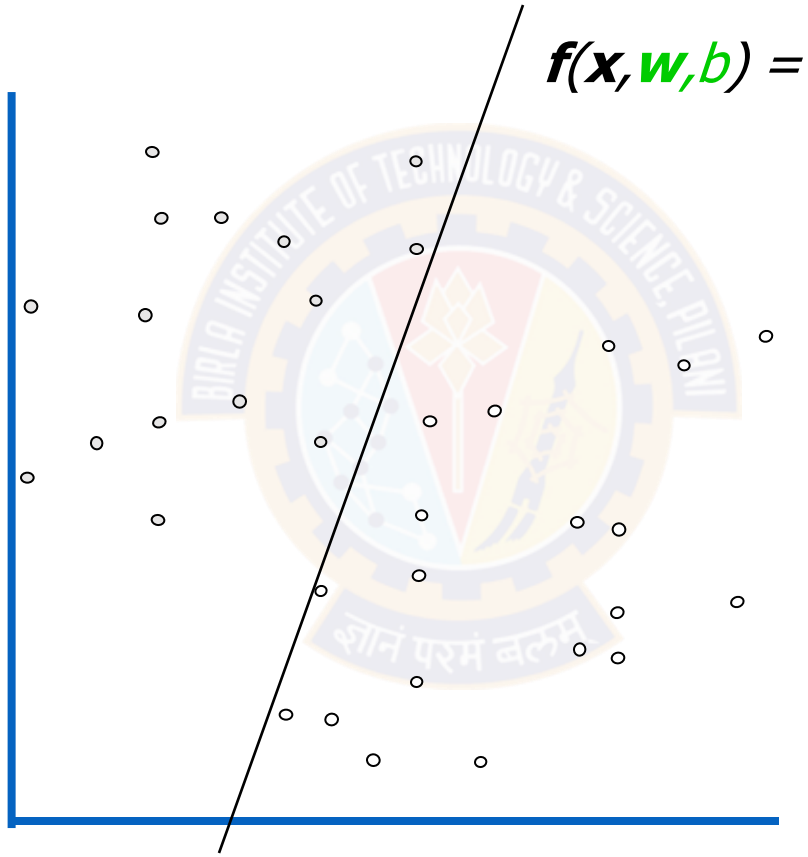
- denotes +1
- denotes -1



How would you classify this data?

# Linear Classifiers

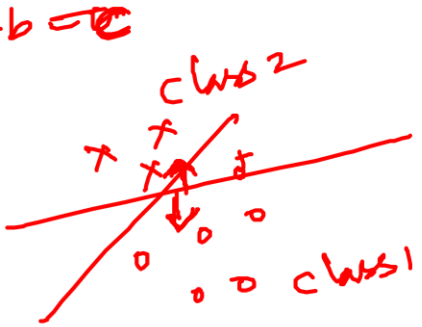
- denotes +1
- denotes -1



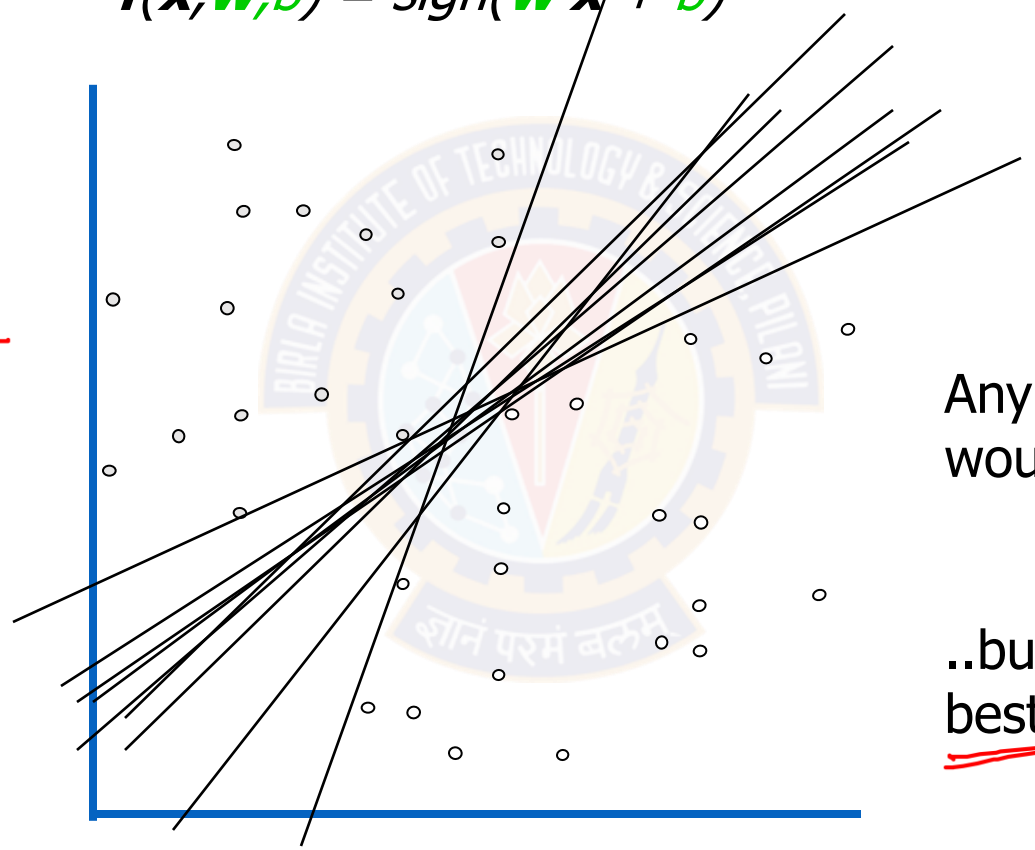

How would you classify this data?

# Linear Classifiers

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$




- denotes +1
- denotes -1



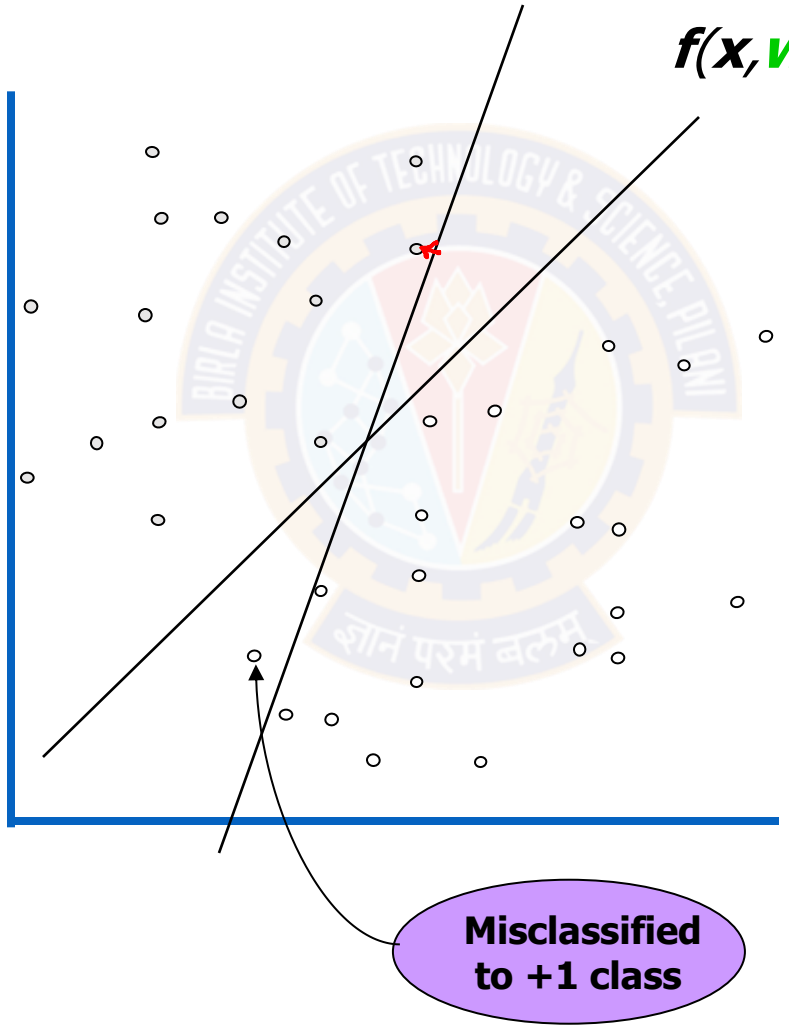
Any of these would be fine..

..but which is best?

# Linear Classifiers

- denotes +1
- denotes -1

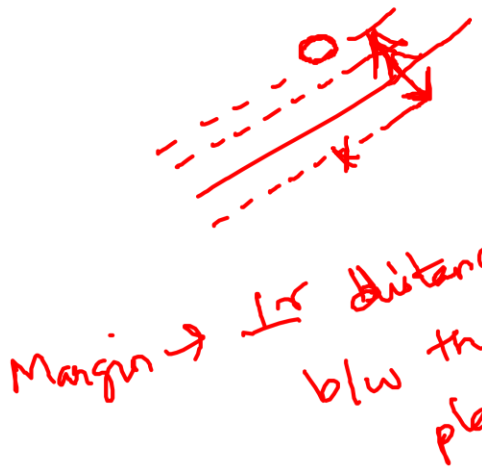
$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$$

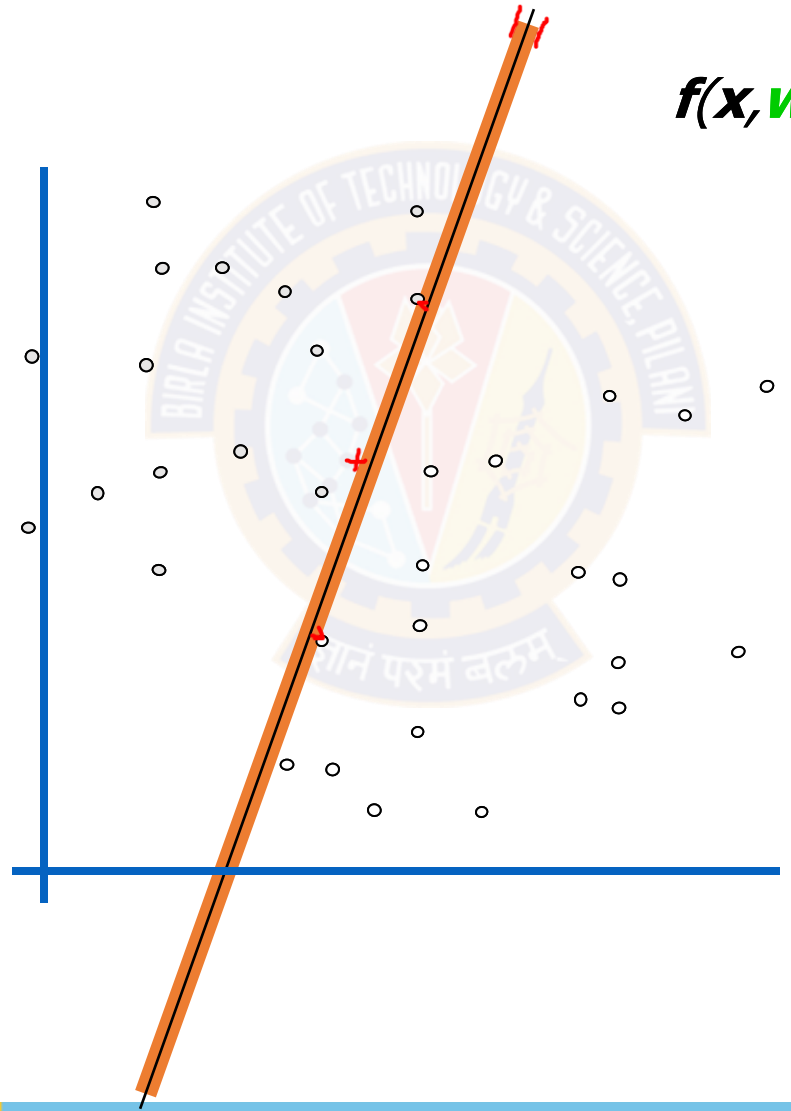



How would you classify this data?

# Linear Classifiers

- denotes +1
- denotes -1



$$f(\boldsymbol{x}, \boldsymbol{w}, b) = sign(\boldsymbol{w}\,\boldsymbol{x} + b)$$

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

# Maximum Margin

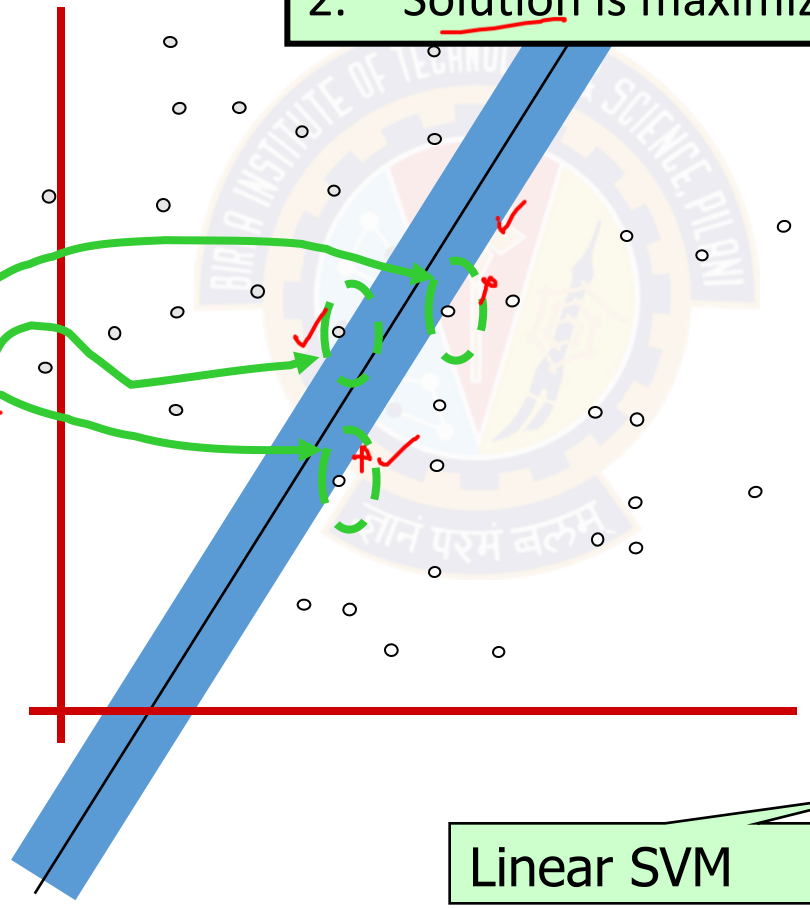
1. If hyperplane is oriented such that it is close to some of the points in your training set, new data may lie on the wrong side of the hyperplane, even if the new points lie close to training examples of the correct class.
2. Solution is maximizing the margin

2D → line
3D → plane

denotes +1

denotes -1

Support Vectors are those datapoints that the margin pushes up against



Linear SVM

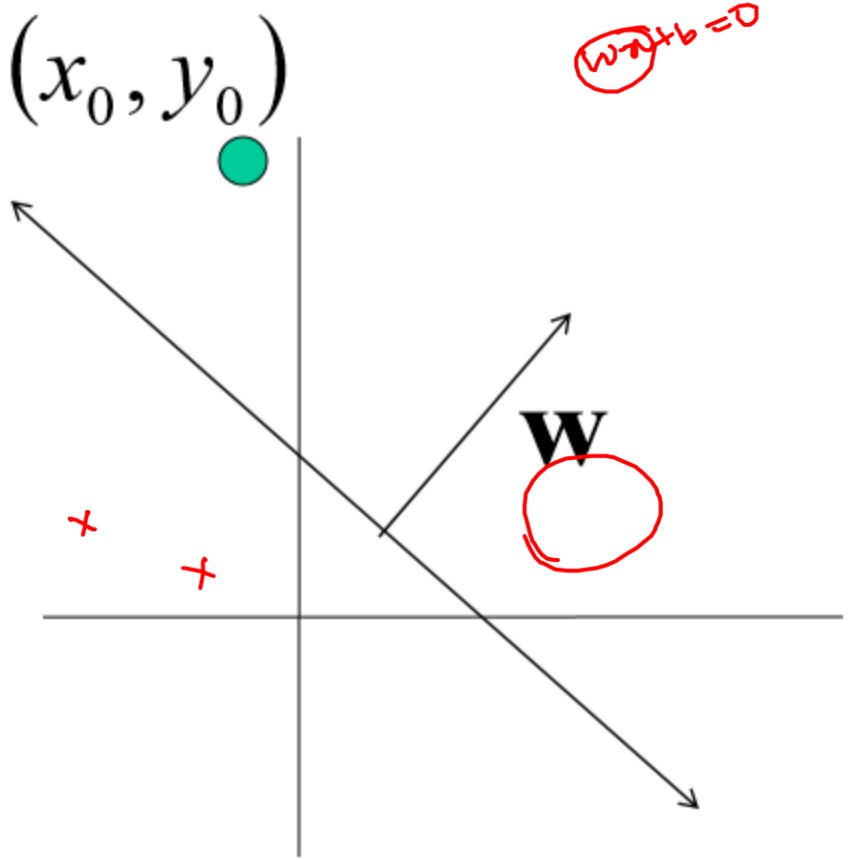
The maximum margin linear classifier is the linear classifier with the, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

# Support Vectors

- Geometric description of SVM is that the max-margin hyperplane is completely determined by those points that lie nearest to it.
- Points that lie on this margin are the support vectors.
- The points of our data set which if removed, would alter the position of the dividing hyperplane

# Line with 2 features: R2




Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

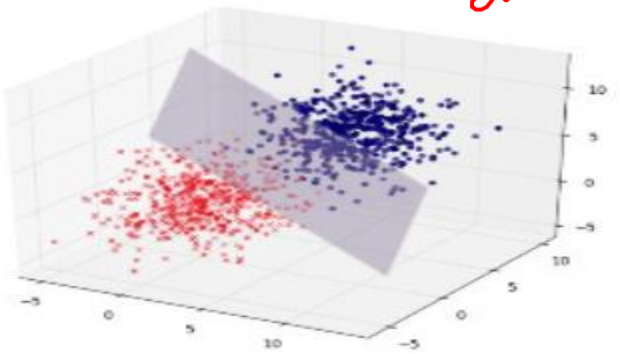
**w·x** controls the **direction/slope** of the decision boundary.
$b$ shifts the boundary **left/right (or up/down)**.

# Example

$$\mathbf{w}^T\mathbf{x} = 0$$

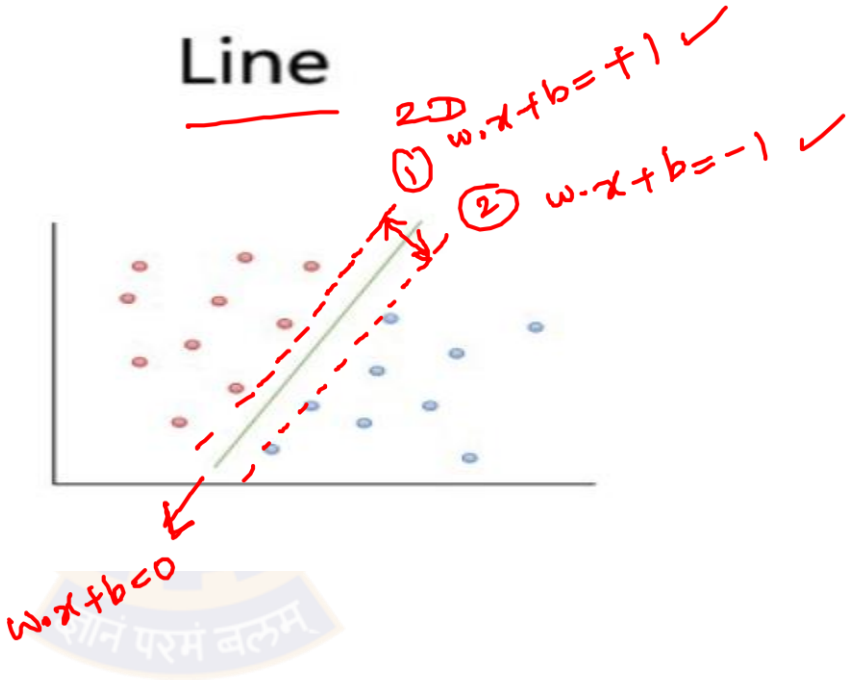
Hyperplane

is higher dimension



$$y = ax + b$$

Line



## 📐 Formula for Perpendicular Distance:

The perpendicular (shortest) distance $D$ from point $(x_0, y_0)$ to the line $ax + by + c = 0$ is:

$$D = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

**Distance Between Parallel Hyperplanes**

We want to calculate the distance between:

- $w^T x + b = 1$
- $w^T x + b = -1$

$\|w\| = \sqrt{a^2+b^2}$

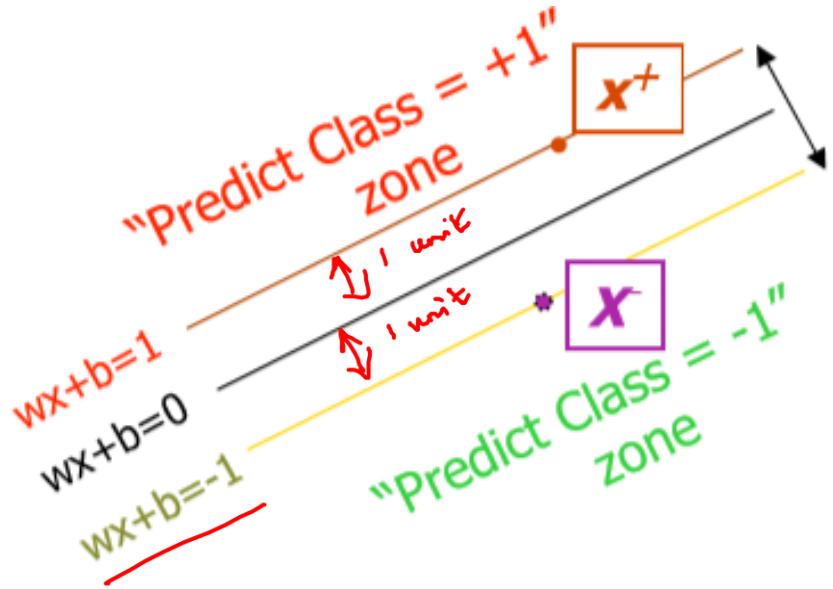
Let's pick any point $x_+$ on the +1 hyperplane and find its perpendicular distance to the –1 hyperplane. Compute the perpendicular (shortest) distance between these two planes using the formula:

$$\text{Distance} = \frac{|w^T x + b - c|}{\|w\|}$$

For the distance between two planes $w^T x + b = a$ and $w^T x + b = b$, this becomes:

$$\text{Distance} = \frac{|a - b|}{\|w\|}$$

# Linear SVM Mathematically




**M**=Margin Width

**Subtracting those equations:**

$$(w \cdot x^+) + b - [(w \cdot x^-) + b] = 1 - (-1)$$

$$w \cdot (x^+ - x^-) = 2$$

+class -ve class

What we know:

- $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$
- $\boldsymbol{w} \cdot (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{\|w\|} = \frac{2}{\|w\|}$$

Width

# Solving the Optimization Problem

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$
$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

*Quadratic optimization problem*:

$$\text{Minimize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{Subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

One constraint for each training point.

Note sign trick.

# Solving the Optimization Problem

Find $\mathbf{w}$ and b such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w^T x_i} + b) \geq 1$

← Primal

- Need to optimize a *quadratic* function subject to *linear inequality* constraints.
- All constraints in SVM are linear
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* $\alpha_i$ is associated with every constraint in the primal problem

**Why Do We Use ||w||²?**

- ||w|| controls the **steepness** of the decision boundary.
- A smaller ||w|| means a **wider margin** (better generalization).
- **Minimizing** $\frac{1}{2} \| w \|^2$ leads to the **maximum margin hyperplane**.

**Geometric Meaning of $b$:**

- $b$ **shifts the hyperplane** up or down (in 2D) or left/right in higher dimensions.
- It **controls how far** the hyperplane is from the **origin**.
- Changing $b$ slides the hyperplane **along the direction perpendicular to** $w$.

# Example:

Suppose we have 2D data points:

Class +1: $x_1 = [2, 2]$ ,$x_2 = [4, 4]$

Class –1: $x_3 = [2, 4]$ ,$x_4 = [4, 6]$

Suppose we find:

$$w = [1, -1]$$
$$b = 0$$

Then the hyperplane is:

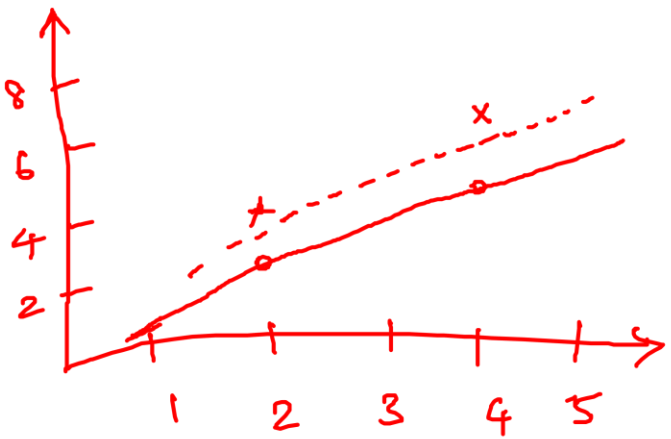
$$x_1 - x_2 = 0 \text{ (i.e., the line } x = y)$$

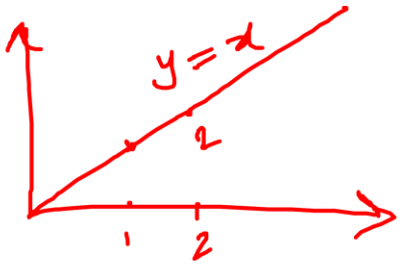
And the squared norm of $w$ is:

$$\| w \|^2 = 1^2 + (-1)^2 = 2$$

So the **margin** is:

$$\frac{2}{\sqrt{2}} = \sqrt{2}$$




If we found another vector $w$ with a **smaller norm**, and it still correctly separated the classes, it would give us a **larger margin**, which is what SVM seeks.

# Solving the Optimization Problem

- The solution involves constructing a *dual problem* where a *Lagrange multiplier* $\alpha_i$ is associated with every constraint in the primary problem:

$$L(w, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \Sigma\, \alpha_i [y_i (\mathbf{w}^T \mathbf{x_i} + b) - 1]$$

- Taking partial derivative with respect to w , $\frac{\partial L}{\partial w} = 0$
  - $w - \Sigma\, \alpha_i y_i x_i = 0$
  - $w = \Sigma\, \alpha_i y_i x_i$ → support vectors
- Taking partial derivative with respect to b, $\frac{\partial L}{\partial b} = 0$
  - $-\Sigma\, \alpha_i y_i = 0$
  - $\Sigma\, \alpha_i y_i = 0$

f → obj. function.
gj → constraint
Lagrange multiplier method:

$f + \lambda g = L$

$\|w\|^2 = w_1^2 + w_2^2 + \cdots + w_n^2 = \sum_{j=1}^{n} w_j^2$

$\frac{\partial L}{\partial w}\left[\|w\|^2\right] = 2w_j$

$f = \frac{1}{2}x^2 + 2y$

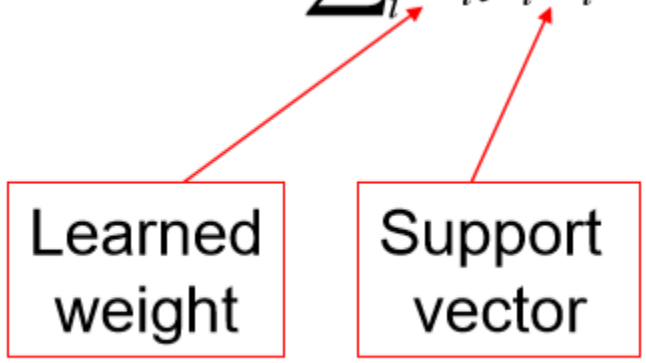
$\frac{\partial f}{\partial x} = \frac{1}{2}(2x) = x$

$\frac{df}{dx} = 0$

$\frac{d^2 f}{dx^2} < 0$

# Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$



# Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

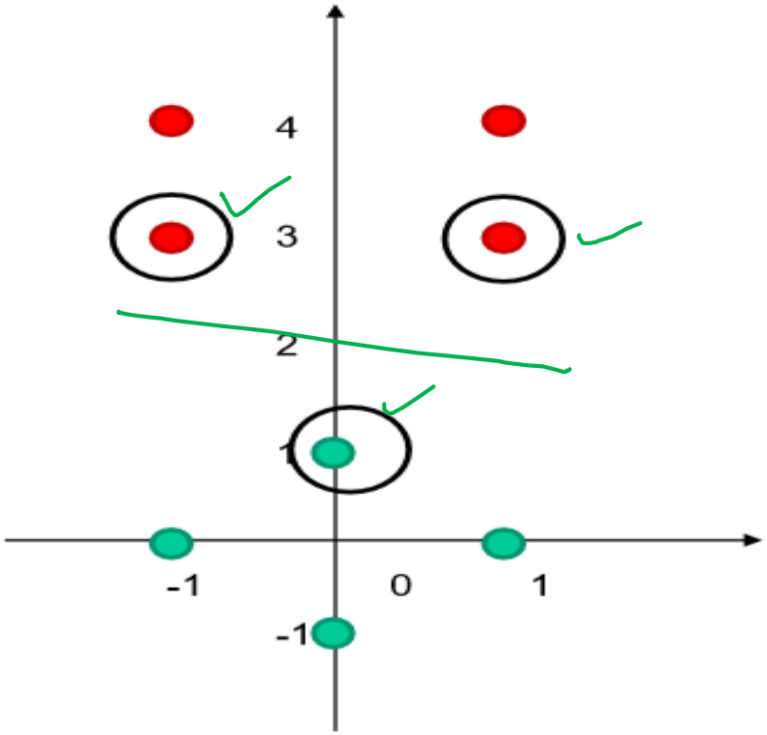
- Classification function:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + \mathrm{b})$$
$$= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

*If f(x) < 0, classify as negative, otherwise classify as positive.*

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

# Example




POS

NEG

= support vectors

Example adapted from Dan Ventura

# Solving for α

- We know that for the support vectors, f(x) = 1 or -1 exactly
- Add a 1 in the feature representation for the bias
- The support vectors have coordinates and labels:
  - x1 = [0 1 1], y1 = -1
  - x2 = [-1 3 1], y2 = +1
  - x3 = [1 3 1], y3 = +1
- Thus we can form the following system of linear equations:

# Solving for α

x1 = [0 1 1], y1 = -1
x2 = [-1 3 1], y2 = +1
x3 = [1 3 1], y3 = +1

- System of linear equations:

α1 y1 dot(x1, x1) + α2 y2 dot(x1, x2) + α3 y3 dot(x1, x3) = y1
α1 y1 dot(x2, x1) + α2 y2 dot(x2, x2) + α3 y3 dot(x2, x3) = y2
α1 y1 dot(x3, x1) + α2 y2 dot(x3, x2) + α3 y3 dot(x3, x3) = y3

-2 * α1 + 4 * α2 + 4 * α3 = -1
-4 * α1 + 11 * α2 + 9 * α3 = +1
-4 * α1 + 9 * α2 + 11 * α3 = +1

- Solution: α1 = 3.5, α2 = 0.75, α3 = 0.75

# Solving for w and b

We know $w = \alpha_1 y_1 x_1 + \ldots + \alpha_N y_N x_N$ where N = # SVs

Thus w = -3.5 * [0 1 1] + 0.75 [-1 3 1] + 0.75 [1 3 1] = [0 1 -2]

Separating out weights and bias, we have: w = [0 1] and b = -2

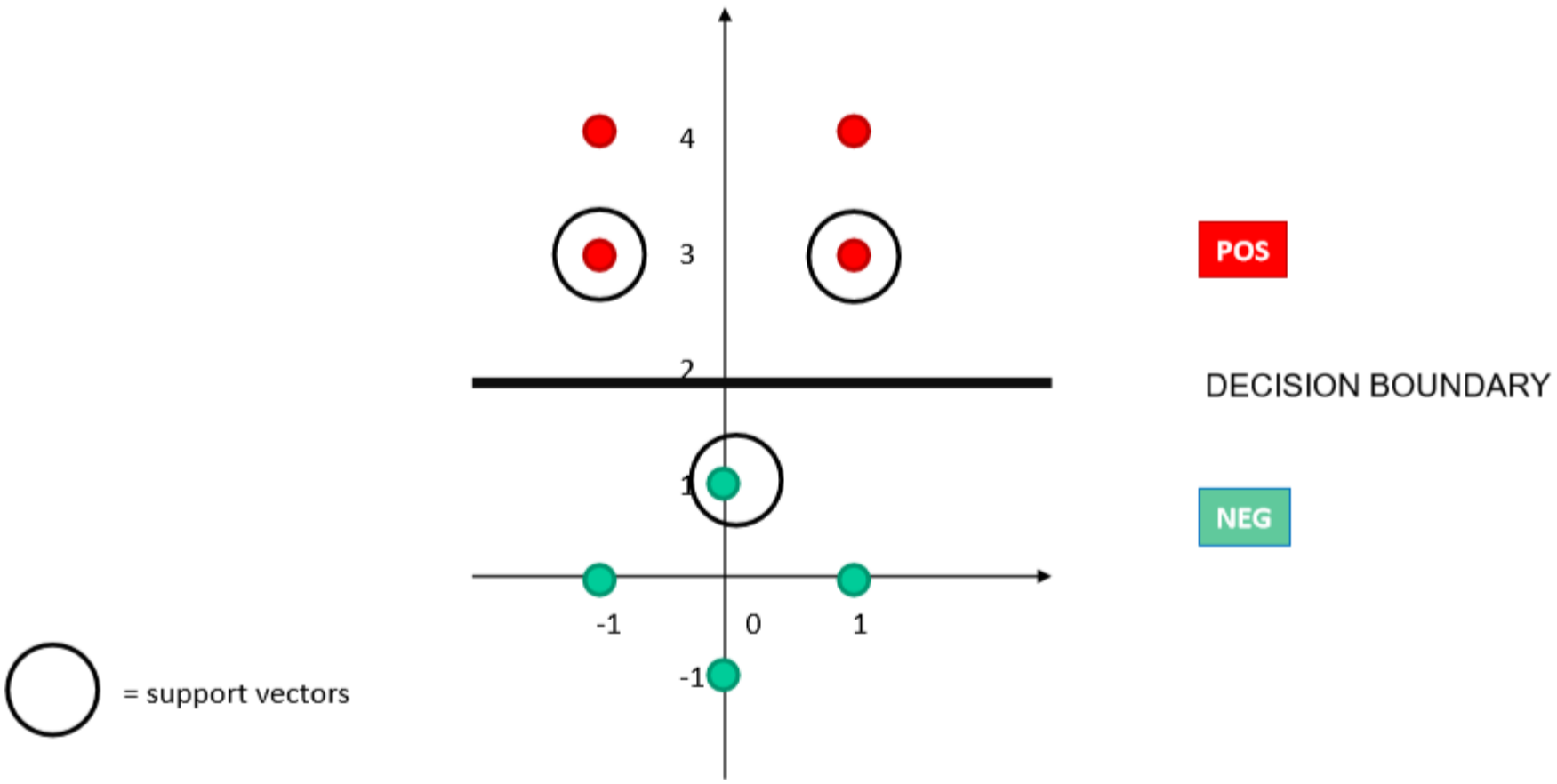
For SVMs, we used this eq for a line: ax + cy + b = 0 where w = [a c]

Thus ax + b = -cy ➔ y = (-a/c) x + (-b/c)

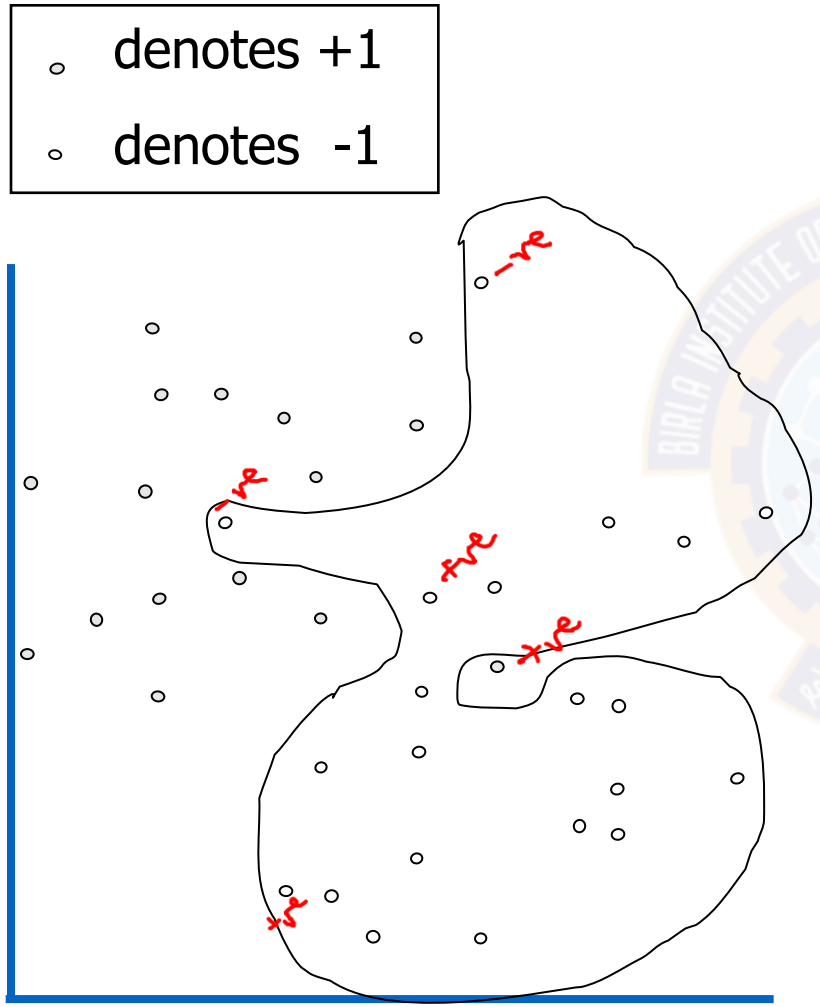
Thus y-intercept is -(-2)/1 = 2

The decision boundary is perpendicular to w and it has slope -0/1 = 0

# Decision boundary



# Dataset with noise




- **Hard Margin: So far we require all data points be classified correctly**
- No training error
- **What if the training set is noisy?**

# Soft Margin Classification

***Slack variables*** $\xi_i$ **can be added to allow misclassification of difficult or noisy examples.**

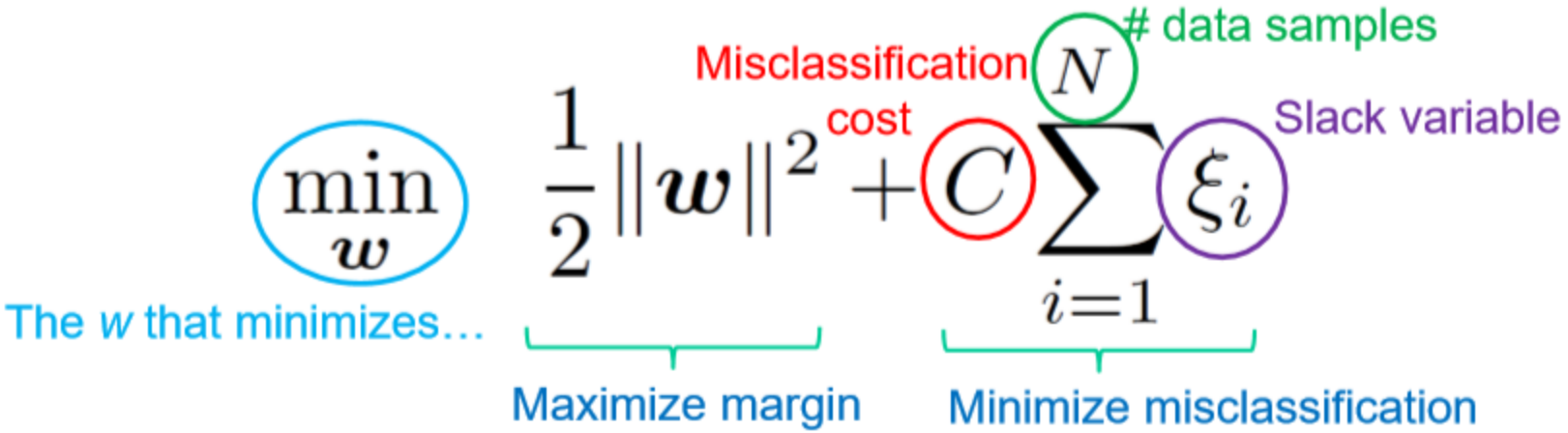
What should our quadratic optimization criterion be?

Minimize $$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

# Slack Variable

- **Slack variable** as giving the classifier some leniency when it comes to moving around points near the **margin**.
- When C is large, larger slacks penalize the objective function of SVM's more than when C is small.

# Soft Margin




$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \quad \forall i = 1, \ldots, N$$

# Hard Margin vs Soft Margin

- **Hard Margin:**

Logic → classifiers are build

Find $\mathbf{w}$ and $b$ such that

$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **Soft Margin incorporating slack variables:**

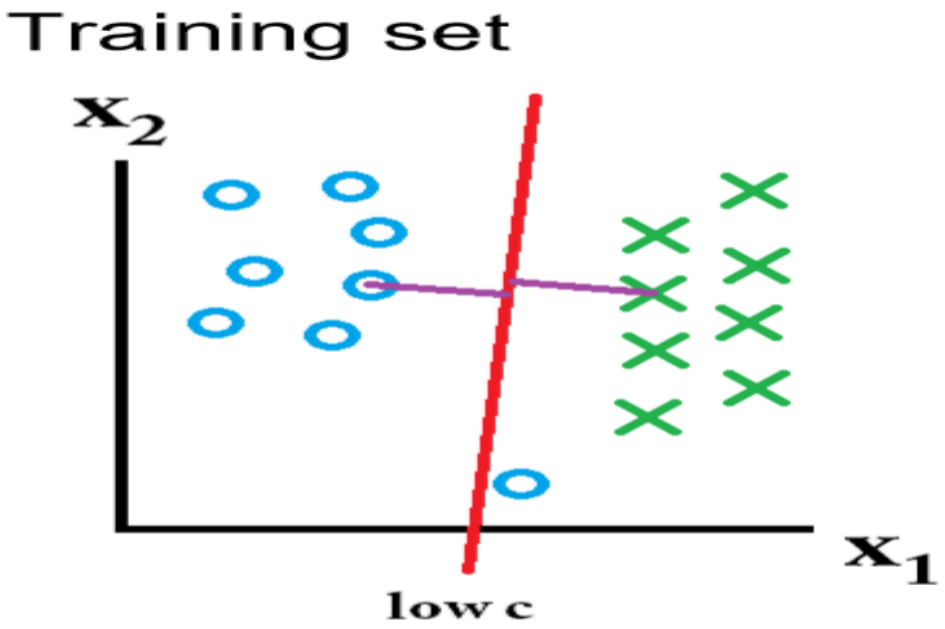
Find $\mathbf{w}$ and $b$ such that

$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

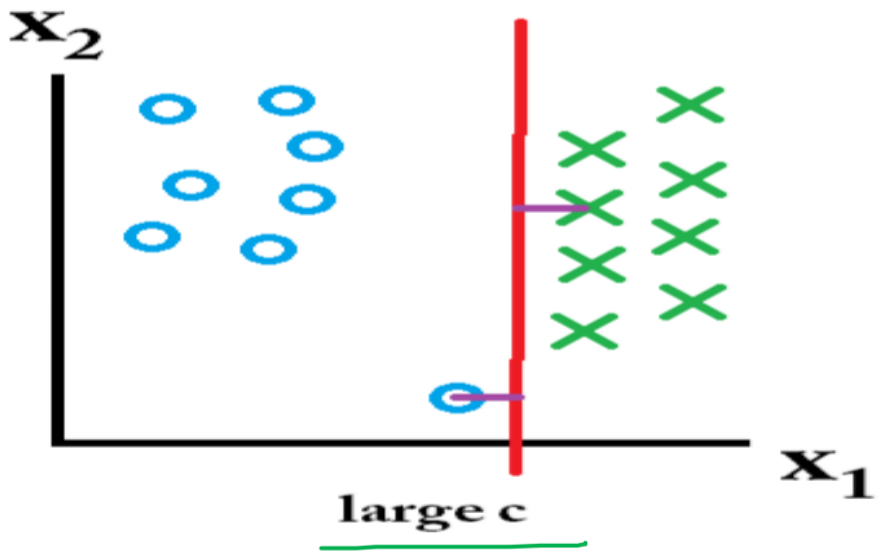
$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all $i$

- **Parameter *C* can be viewed as a way to control overfitting.**

# Effect of Margin size v/s misclassification cost

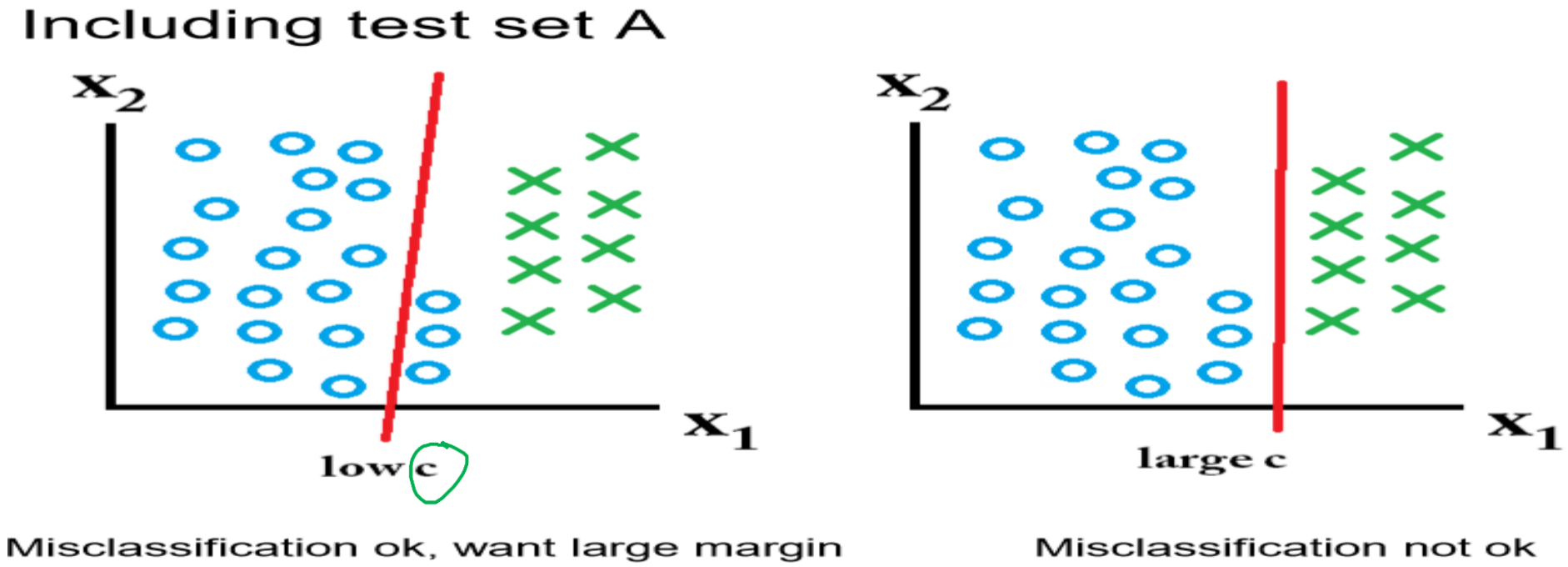



Misclassification ok, want large margin




Misclassification not ok

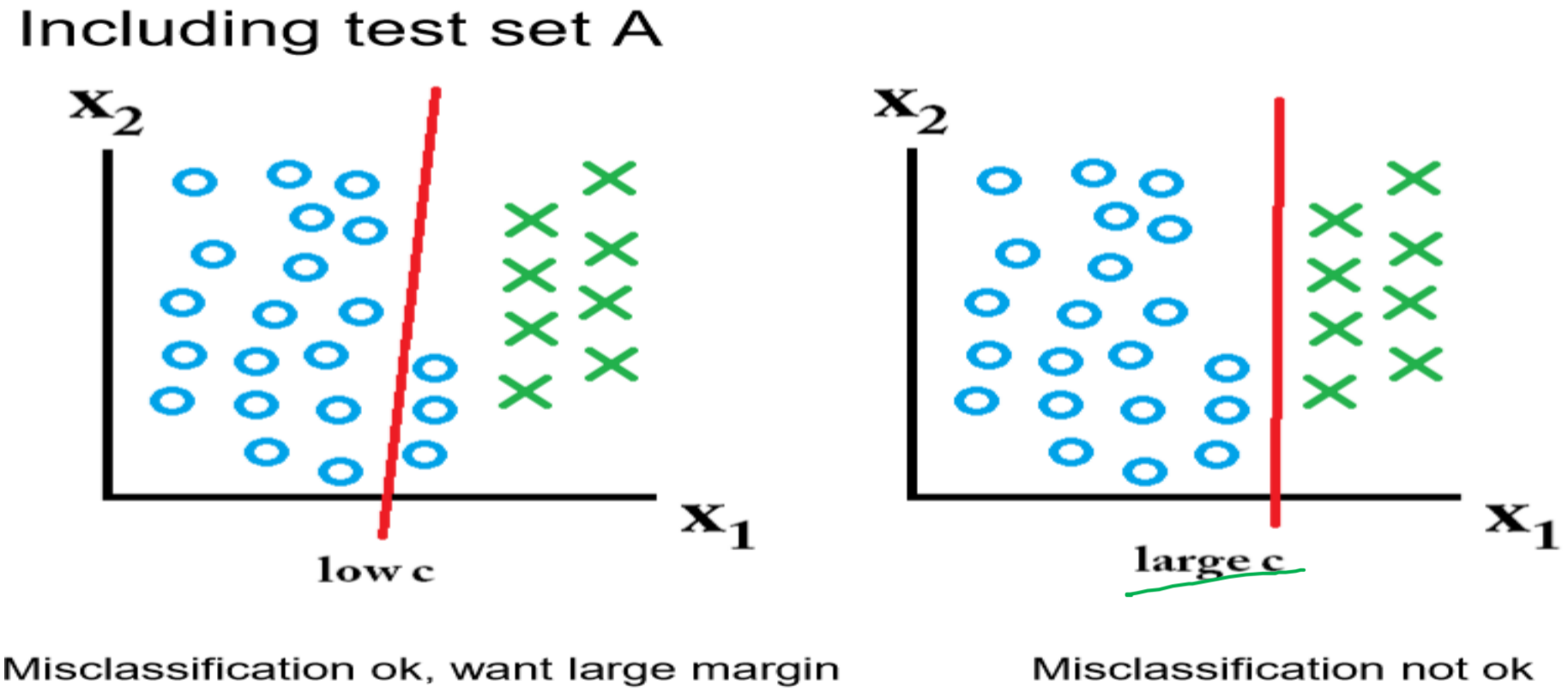
# Effect of Margin size v/s misclassification cost




Misclassification ok, want large margin

Misclassification not ok

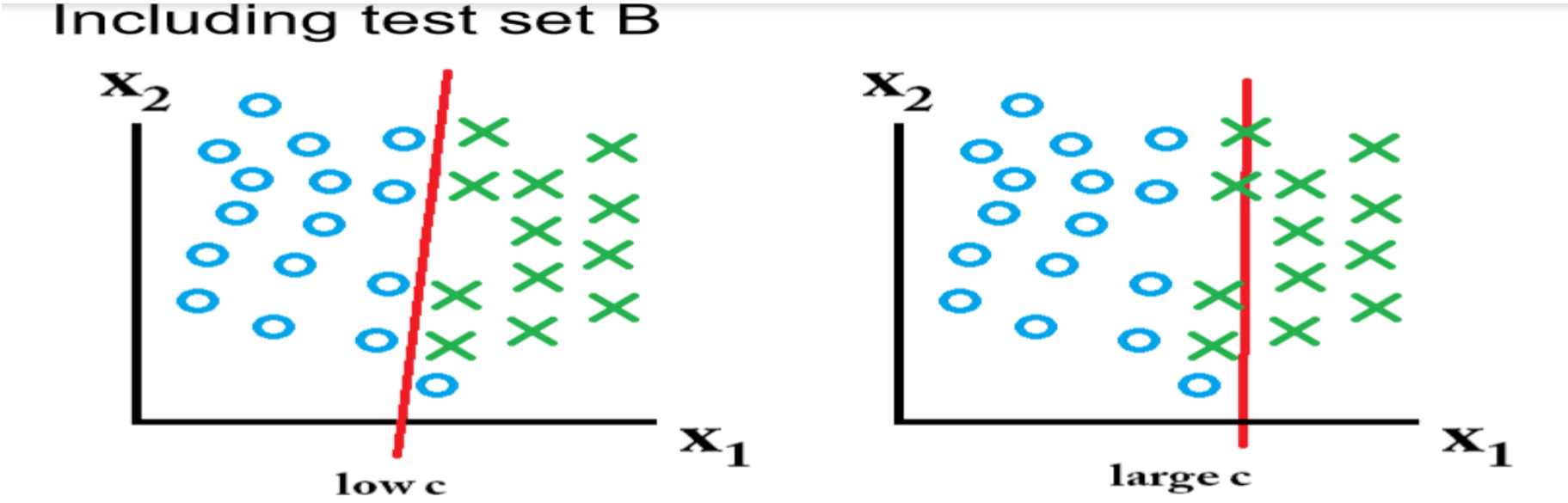
# Effect of Margin size v/s misclassification cost




Misclassification ok, want large margin

Misclassification not ok

# Effect of Margin size v/s misclassification cost




Misclassification ok, want large margin

Misclassification not ok

# Linear SVMs: Overview

- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $x_i$ are support vectors with non-zero Lagrangian multipliers $\alpha_i$.

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

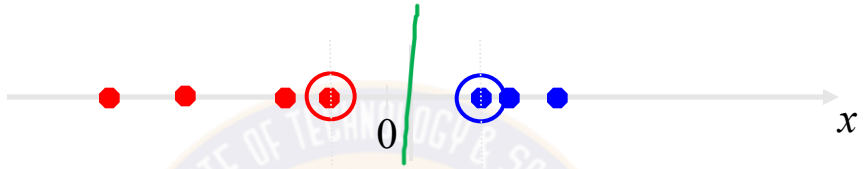
→ $\alpha_i$ → here

$f + \lambda g = L$

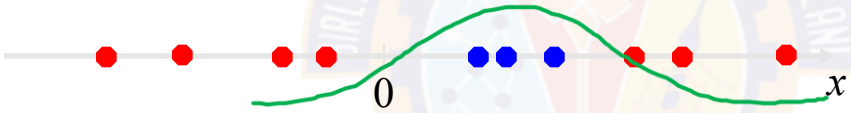
# Support Vector Machines in overlapping class distributions & Kernels

# Non-linear SVMs

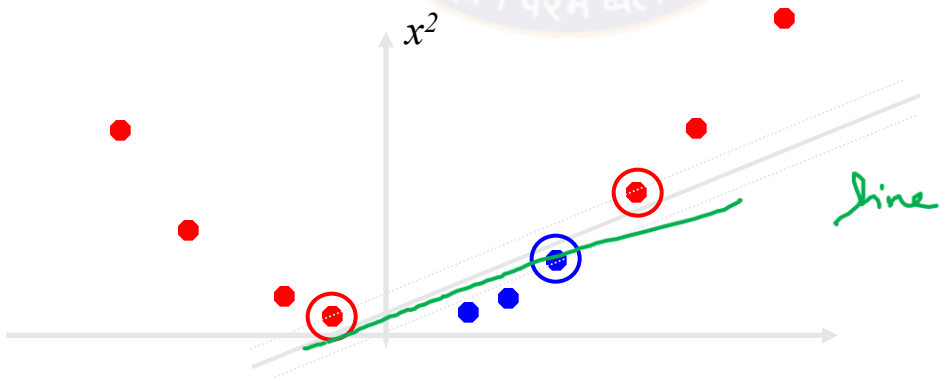
- Datasets that are linearly separable with some noise soft margin work out great:




- But what are we going to do if the dataset is just too hard?




- How about… mapping data to a higher-dimensional space:



# The "Kernel Trick"

- The linear classifier relies on dot product between vectors
  - $x_i^T \cdot x_j$

$w \cdot x + b$

- If every data point is mapped into high-dimensional space via some transformation $\Phi: x \rightarrow \phi(x)$, the dot product becomes:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

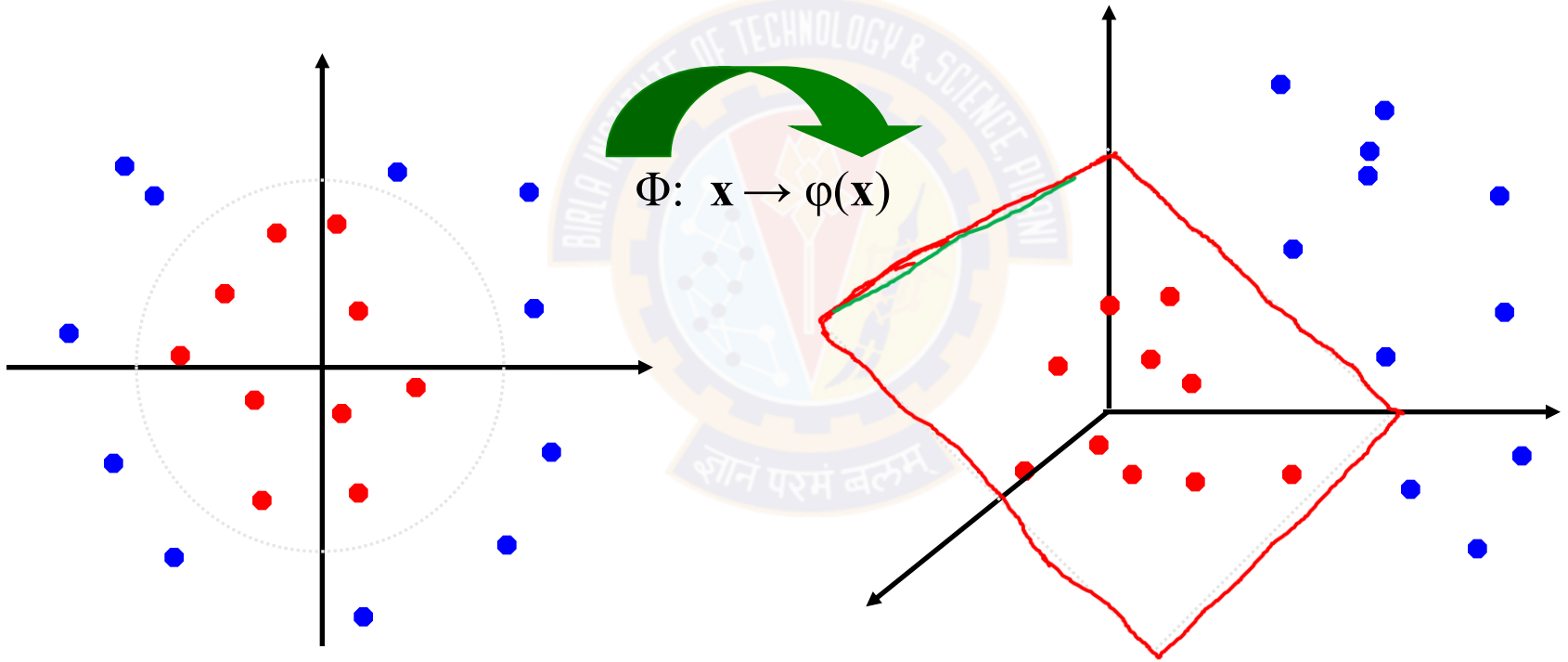
# SVM Kernels

- SVM algorithms use a set of mathematical functions that are defined as the kernel.
- Function of kernel is to take data as input and transform it into the required form.
- Different SVM algorithms use different types of kernel functions.

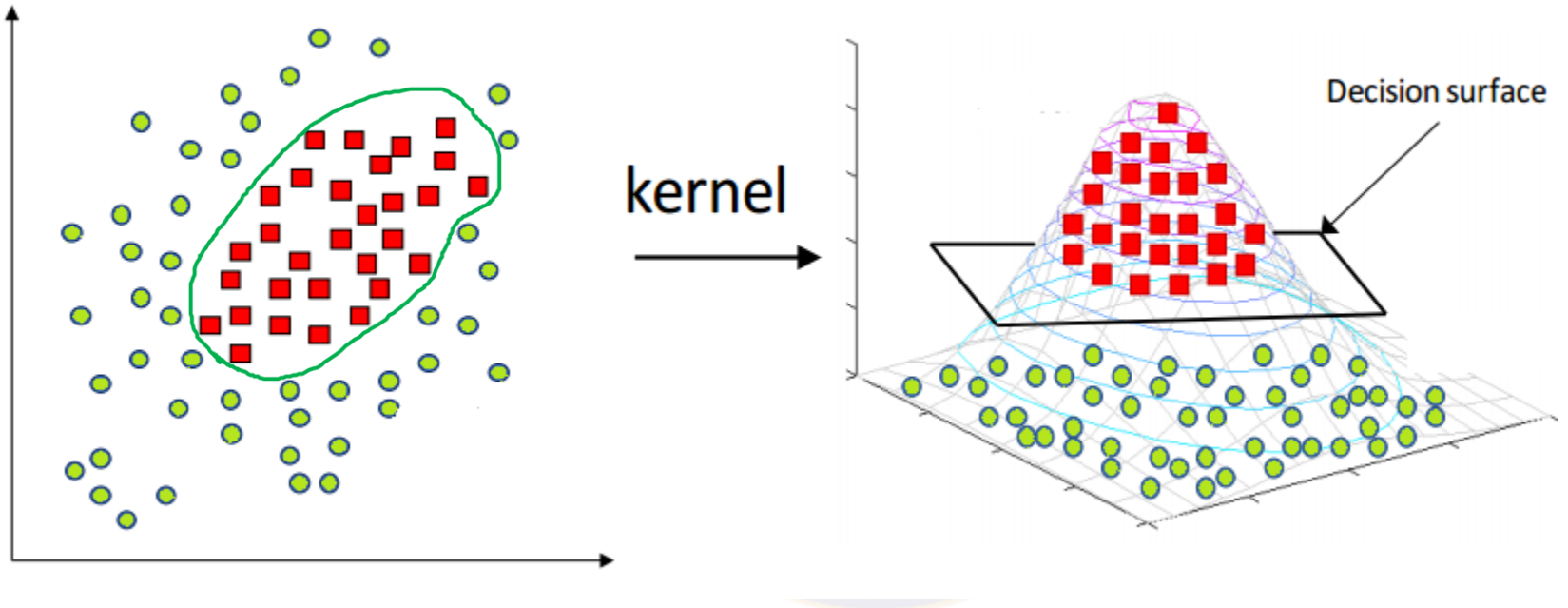
  Example *linear, nonlinear, polynomial, and sigmoid etc.*

# Non-linear SVMs: Feature spaces

- **General idea**: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable



# Kernel Trick



# Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
- Polynomial of power $p$: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$

$(1+xy)^2 = (1+x^2y^2+2xy)$

- Gaussian (radial-basis function network):

normal curve

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x}_i,\mathbf{x}_j)= \tanh(\beta_0\mathbf{x}_i^T\mathbf{x}_j + \beta_1)$

# Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

# Kernel Trick

Maximize

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{< \vec{x_i}, \vec{x_j} >}$$

Subject to $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$

Non-linear SVM

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{K(\vec{x_i}, \vec{x_j})}$$

Lagrange ?? Construction

Linear SVM: (without mapping)

w. x

$$f(\vec{x}) = < \vec{w}, \vec{x} > + b$$

$$= \sum_i \alpha_i y_i \boxed{< \vec{x_i}, \vec{x} >} + b$$

Non-linear SVM: w could be infinite dimensional

$$f(\vec{x}) = \sum_i \alpha_i y_i \boxed{K(\vec{x_i}, \vec{x})} + b$$

# Multi-Class Problem

- **Instead of just two classes , we now have C classes**
  - Examples: predict which movie genre a viewer likes best
  - Possible answers : Action, Drama , thriller
- **Two approaches**

  one – vs – all

  one –vs –one
- **One – vs – all(a.k.a .one – vs –others)**
  - Train C classifiers
  - In each , pos = data from class I, neg = data from classes other than I
  - The class with the most confident prediction wins
- Example
  - You have 4 classes , train 4 classifiers

    1 vs others : score 3.5

    2 vs others : score 6.2

    3 vs others : score 1.4

    4 vs others : score 5.5
  - Final prediction: class 2

# Multi-Class Problem

- One – vs – one (a.k.a.all – vs – all)
- Train C binary classifiers(all pairs of classes)
- They all vote for the label
- Example
  - You have 4 classes , then train 6 classifiers
  - 1 vs 2,1 vs 3,1 vs 4,2 vs 3,2 vs 4,3 vs 4
  - Votes: 1,1,4,2,4,4
  - Final prediction class 4

A, B, C, D

# SVMs: Pros and cons

**Pros**

- Kernel-based framework is very powerful, flexible
- Often a sparse set of support vectors – compact at test time
- Work very well in practice, even with very small training sample sizes
- Solution can be formulated as a quadratic program (next time)
- Many publicly available SVM packages: e.g. LIBSVM, LIBLINEAR, SVMLight

**Cons**

- Can be tricky to select best kernel function for a problem
- Computation, memory
  - At training time, must compute kernel values for all example pairs
  - Learning can take a very long time for large-scale problems

# Good Web References for SVM

- **Text categorization with Support Vector Machines:learning with many relevant features** - T. Joachims, ECML
- **A Tutorial on Support Vector Machines for Pattern Recognition**, Kluwer Academic Publishers - Christopher J.C. Burges
- http://www.cs.utexas.edu/users/mooney/cs391L/
- https://www.coursera.org/learn/machine-learning/home/week/7
- https://data-flair.training/blogs/svm-kernel-functions/
- MIT 6.034 Artificial Intelligence, Fall 2010
- https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior
- https://www.sciencedirect.com/science/article/abs/pii/S089360800600279 6
- https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be
- Radial basis kernel

# Thank you